REMARKS

The above Amendments and these Remarks are in reply to the Office Action dated September 13, 2011. Currently, claims 1-2, 5-13, 17-23, 28-30, 32-35, 37-42, 44-47, 51 and 52 are pending. Applicants have amended claims 1-2, 5-11, 13, 17-20, 23, 28-30, 32-35, 37-38, 41-42, 46, 51-52. Claim 3 have been cancelled, without prejudice. No new matter has been added as a result of the claim amendments.

Applicants respectfully request reconsideration of claims 1-2, 5-13, 17-23, 28-30, 32-35, 37-42, 44-47, 51 and 52.

Examiner Interview Summary

On December 7, 2011, the Applicants' representative, Ronald M. Pomerenke, conducted a telephonic interview with Examiner Zheng Wei. The Applicants thank the Examiner for granting this interview. A proposed amendment to claim 1 was discussed. *Berkley et al.* (hereinafter "*Berkley*"); U.S. Patent No. 6,738,965 to *Webster* (hereinafter "*Webster*"); "*Call Graph Construction in Object-Oriented Languages*," by *Grove et al.* (hereinafter "*Grove*"); and "*A Framework for Call Graph Construction Algorithms"* (David Grove and Craig Chambers) (hereinafter "*Grove and Chambers*") were discussed. No definitive agreements were reached.

Rejection of Claims 1-3, 5-8, 10, 12-13, 17-18, 20, 39-42, 44, 47 and 51-52 Under 35 U.S.C. §103(a)

Claims 1-3, 5-8, 10, 12-13, 17-18, 20, 39-42, 44, 47 and 51-52 were rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,351,843 to *Berkley et al.* (hereinafter "*Berkley*"), in view of U.S. Patent No. 6,738,965 to *Webster* (hereinafter "*Webster*"), in further view of "Call Graph Construction in Object-Oriented Languages," by *Grove et al.* (hereinafter "*Grove*"). The rejection is respectfully traversed for the following reasons.

Claim 1

> A process for monitoring, comprising:
> accessing a method;

automatically determining whether to modify said method, said step of automatically determining whether to modify said method includes automatically determining whether said method calls another method and if said method has an access level that satisfies a criterion; and

modifying said method for a particular purpose only if said method calls another method and said access level satisfies the criterion.

Applicants respectfully assert that *Berkley* in view of *Webster* in further view of *Grove* fails to disclose, "*modifying said method for a particular purpose only if said method calls another method and said access level satisfies the criterion*," as claimed.

Berkley by itself does not disclose these claim limitations. Moreover, there is no reason to modify *Berkley* to arrive at these claim limitations. *Berkley* discloses monitoring classes in an object-oriented environment. Applicants respectfully assert that *Berkley* cannot be modified to trace on a method by method basis without **destroying *Berkley* as a reference**. Stated another way, *Berkley* cannot be modified to trace on a method by method basis without **changing the fundamental principle of operation of *Berkley***. Therefore, at the time of Applicants' invention, it would not have been obvious to one of ordinary skill in the art to modify *Berkley* to arrive at the claimed invention.

As noted, *Berkley* discloses monitoring classes in an object-oriented environment. A class may include one or more objects, and the class defines how an object is implemented. Objects are instances of classes (col. 4, lines 61-62). Each object may include one or more methods, as shown in Figure 1. Therefore, a class may contain one or more objects, which each may contain one or more methods. As shown in Figure 1, a method may or may not call another method. For example, in Object 1, which is part of a class, Method A calls Method B. However, Method D does not call any other method.

Berkley discloses monitoring methods by class as follows. If a user wishes to monitor a particular method within a program, the user can change configuration settings associated with the program to specify or activate the class of the method that the user wishes to monitor. "[C]onfiguration settings 300 received as part of a program's execution code are modified at runtime to add a setting to specify (as one example) tracing for a desired class of the executable 310, thereby

producing new configuration settings 320" (Berkley Figure 5 and col. 6, lines 52-53). *Berkley* further explains that:

> the new configuration settings 330 are then employed with the existing application executable 340 to run the application executable 350. The object runtime will query the configuration settings and when a trace is active for a given class, will dynamically insert the trace class into the inheritance hierarchy for that class… Runtime will then see the trace class within the hierarchy and setup for a trace 360 (Berkley col. 6, line 64 – col. 7, line 4).

For example, for a class named 'class 1' that is specified in the configuration settings, the trace is set up through "redirection stubs [that] are created [to] trace entry and exit methods around each target method within class 1" (*Berkley* col. 7, lines 61-63). A trace is active for a given class when the user specifies the class in the configuration settings. The example for activating classes in column 7, lines 20-26 of *Berkley* shows:

> configuration settings used to set up a desired tracing environment. A configuration variable is initialized with the name(s) of the class(es) whose methods should be traced:
> [TraceOptions]
> ClassTrace = Class1, Class2, Class3, Class4

In the above example, Class1, Class2, Class3, and Class4 are set active by specifying these classes in the ClassTrace list. **All of the methods** in these classes will be traced. Using Figure 1 of *Berkley* as an example, if Object 1 was contained within Class 1, which is specified in the ClassTrace list above, all of the methods contained within Object 1 would be modified for tracing, even Method D which does not call any other method. Since all of the methods are traced, Applicants respectfully asset that *Berkley* cannot be modified to arrive at the claimed invention without destroying *Berkley* as a reference.

Stated another way to attempt to modify *Berkley* to arrive at the claimed invention would fundamentally alter the principle of operation of *Berkley*. One of ordinary skill in the art would not have had a reason to modify *Berkley* given this need to fundamentally alter the principle of operation.

As the foregoing clearly states, *Berkley* discloses that tracing performed by activating **classes**. *Berkley* does not disclose a technique for a user to specify a particular method to trace. Therefore, *Berkley* cannot be modified to arrive at, "*modifying said method for a particular purpose only if said method calls another method and said access level satisfies the criterion*," as claimed.

Webster is concerned with providing a system that allows a user to specify methods to trace, and to specify what information about those methods the user would like. However, the system that *Webster* has taught does not disclose the claim limitations. Moreover, there is simply no reason to modify *Webster's* system to arrive at the claimed invention or to combine *Webster's* teachings with another reference to arrive at the claimed invention.

Webster discloses that a file is provided that *lists* methods to trace. Applicants first note that *Webster* teaches that a user provides the file. Therefore, in *Webster* the **user has control** over specifically what methods to trace. The claim limitations are for **automatically** determining whether to modify said method. Automating the process of producing the list of methods and classes destroys *Webster* as a reference. Stated another way, it would fundamentally change the principle of operation of *Webster* to automate the process of allowing providing the list. Note that the user would no longer have control to specify what methods to trace, if this change were made to *Webster*.

Furthermore, *Webster* provides no teaching or suggestion to put the following into the list of methods to trace: "whether said method calls another method and if said method has an access level that satisfies a criterion." *Webster* does not in any way teach or suggest either of these, much less the combination. *Webster* does not teach or suggest that a method that calls another method should be added to the list of methods to trace. *Webster* does not teach or suggest that has an access level that satisfies a criterion should be added to the list of methods to trace.

Moreover, since neither *Webster* nor *Berkley* disclose, "automatically determining whether to modify said method, said step of automatically determining whether to modify said method includes automatically determining whether said method calls another method and if said method has an access level that satisfies a criterion," the combination does not disclose these limitations.

Grove discloses that it is difficult to accurately construct call graphs. Applicants refer to "*A Framework for Call Graph Construction Algorithms* (David Grove and Craig Chambers) (herein "*Grove and Chambers*"), which states on page 686 that in object oriented languages and languages with function values, the target of a call cannot always be precisely determined solely by an examination of the source code of the call site. *Grove and Chambers* states that an inter-procedural analysis may need to be performed, which actually requires that a call graph be built prior to the analysis being performed. Thus, *Grove and Chambers* states that this circularity needs to be resolved for "higher level languages" (which *Grove and Chambers* defines as object oriented languages and functional languages.

Applicants have recited a solution in claim 1 that does not require that a call graph be constructed. Rather, the Applicants have discovered that a much simpler solution is possible.

Moreover, Applicants respectfully assert that one of ordinary skill in the art having *Grove* (or *Grove and Chambers*) in front of them would not find it obvious or have any reason to modify *Berkley* in view of *Webster* to arrive at the claim limitations. Applicants respectfully assert that all *Grove* discloses or suggests is a technique for solving the complexities of determining call graphs. Even if *Grove* was used to produce a call graph, which is then presented to the user in *Webster*, the Applicants' claimed invention still does not present itself to the person of ordinary skill in the art. Note that even if, for the sake of argument, the user were to add only methods that are at the top of the call graph to the list of methods to analyze in *Webster*, ***the claim limitations are not met***.

Applicants respectfully assert that there is nothing in *Grove,* even in view of *Berkley* in view of *Webster* that discloses *modifying said method for a particular purpose only if said method calls another method and said access level satisfies the criterion.* For example, simply presenting a call graph in no way supplies what is missing in *Berkley* in view of *Webster.* As noted, *Berkley* discloses monitoring classes in an object-oriented environment. Merely teaching how to construct a call graph does not lead one to the solution of what methods are modified in claim 1. As noted, *Webster* discloses allowing a user to provide a list of methods to trace. Merely teaching how to construct a call graph does not lead one to the solution of **that a method should be added to the list** *only if said method calls another method and said access level satisfies the criterion*.

Applicants note that they have taught a fundamentally different technique than any combination of *Berkley*, *Webster*, and *Grove*. What Applicants have recited in claim 1 does not to use brute force to determine a call graph (which may be inaccurate anyway). Instead, claim 1 recites an elegant solution that modifies a method for a particular purpose *only if said method calls another method and said access level satisfies the criterion*.

For all of the foregoing reasons, Berkley in view of Webster in further view of Grove fails to disclose, "*modifying said method for a particular purpose only if said method calls another method and said access level satisfies the criterion*," as claimed.

Claim 13

> A process for monitoring, comprising:
> automatically determining which methods of a set of methods call one or more other methods and are synthetic; and
> using a first tracing mechanism for said *methods that call one or more other methods and are not synthetic* without using said first tracing mechanism for methods that do not call one or more other methods or are synthetic.

Applicants respectfully assert that *Berkley* in view of *Webster* in further view of *Grove* fails to disclose, "*using a first tracing mechanism for said methods that call one or more other methods and are not synthetic without using said first tracing mechanism for methods that do not call one or more other methods or are synthetic*," as claimed.

As stated in paragraph 37 of the instant specification, "a synthetic method is a method that does not appear in the source code. *Berkley* by itself does not disclose these claim limitations pertaining to tracing **based on whether a method is synthetic**. Moreover, claim 13 has the additional limitation of *methods that call one or more other methods*. Thus, first tracing mechanism is used for said *methods that call one or more other methods* ***and*** *are not synthetic*.

Moreover, it would not be obvious to use a first tracing mechanism for methods that are both synthetic **and** call one or more other methods, even if synthetic methods are known in JAVA programming techniques. First of all, even if all synthetic methods were to be traced (or all not

traced), the claim limitations would not be met. Tracing all methods that are not synthetic would trace those that **do not** call one or more methods. Thus, what the Office Action appears to suggest would be obvious (tracing methods that are not synthetic) does not in fact meet the claim limitations. As Applicants have noted above, Applicants have solved a problem of tracing **too many** methods. Moreover, Applicants have created a solution that is both elegant and simple to implement.

Likewise, *Webster* in further view of *Grove* do not teach these claim limitations. As noted above, *Webster* discloses that a user can provide file that *lists* methods to trace. Applicants first note that *Webster* teaches that a user provides the file. Therefore, in *Webster* the user determines whether to trace the method. The claim limitations are for **automatically** determining whether to modify said method. Moreover, automating the process of producing the list of methods and classes destroys *Webster* as a reference. Stated another way, it would fundamentally change the principle of operation of *Webster* to automate the process of the user providing the list.

Furthermore, *Webster* provides no teaching or suggestion to put the following into the list of methods to trace "methods that call one or more other methods and are not synthetic." *Webster* does not in any way teach or suggest either of these, much less the combination. *Webster* does not teach or suggest that a method that calls another method should be added to the list of methods to trace. *Webster* does not teach or suggest that methods that are not synthetic should be traced.

Moreover, *Grove*'s discussion on constructing call graphs does not remedy the foregoing deficiencies.

Moreover, there is no reason to modify *Berkley* to arrive at these claim limitations. *Berkley* discloses monitoring classes in an object-oriented environment. Applicants respectfully asset that *Berkley* cannot be modified to trace on a method by method basis without **destroying *Berkley* as a reference**. Stated another way, *Berkley* cannot be modified to trace on a method by method basis without **changing the fundamental principle of operation of *Berkley***. Therefore, at the time of Applicants' invention, it would not have been obvious to one of ordinary skill in the art to modify *Berkley* to arrive at the claimed invention.

Applicants note that they have taught a fundamentally different technique than any combination of *Berkley*, *Webster*, and *Grove*. Applicant's invention recited in claim 13 is not to use brute force to determine a call graph (which may be inaccurate anyway). Instead, claim 13 recites an elegant solution of "*using a first tracing mechanism for said methods that call one or more other methods and are not synthetic without using said first tracing mechanism for methods that do not call one or more other methods or are synthetic.*"

For all of the foregoing reasons, Berkley in view of Webster in further view of Grove fails to disclose, "*using a first tracing mechanism for said methods that call one or more other methods and are not synthetic without using said first tracing mechanism for methods that do not call one or more other methods or are synthetic*," as claimed.

Claim 39

Applicants respectfully assert that *Berkley* in view of *Webster* in further view of *Grove* fails to disclose, "*means for tracing said method for a particular purpose only if said method calls another method, said method can be called by a sufficient scope of one or more other methods, and said method is not a synthetic method*," as claimed.

As stated in paragraph 37 of the instant specification, "a synthetic method is a method that does not appear in the source code. *Berkley* by itself does not disclose these claim limitations pertaining to tracing **based on whether a method is synthetic**. Moreover, claim 39 has the additional limitation of *methods that call one or more other methods*. Moreover, claim 39 has the additional limitation of *said method can be called by a sufficient scope of one or more other methods*. Thus, claim 37 recites means for tracing said method for a particular purpose only if said method calls another method, said method can be called by a sufficient scope of one or more other methods, **and** said method is not a synthetic method. Note that this, in effect, gives the **intersection** of these three elements. That is, a method must meet all three criteria to be traced. This limits the

methods that are traced in an intelligent, yet elegant manner.

Moreover, it would not be obvious to have the claimed means for tracing methods that are both synthetic **and** call one or more other methods, even if synthetic methods are known in JAVA programming techniques. First of all, even all if methods that are not synthetic were to be traced, the claim limitations would not be met. Tracing all "not synthetic methods" would trace those that **do not** call one or more methods. Thus, too many methods would be traced.

Moreover, even if it is known that methods in the JAVA programming language **have** a scope, this in no way teaches or makes it obvious to trace, as claimed. Even if, for the sake of argument, all methods of a sufficient scope were traced, this would trace too many methods. This would trace methods of sufficient scope that did not call another method. Therefore, too many methods would be traced. It would also trace methods that were synthetic, which again is too many. Moreover, simply knowing that the concept of calling scope exists does not teach or suggest tracing based thereon.

Furthermore, *Webster* in further view of *Grove* do not remedy these deficiencies. As noted above, *Webster* discloses that a user can provide file that *lists* methods to trace. *Webster* does not teach or suggest that a method that calls another method should be added to the list of methods to trace. *Webster* does not teach or suggest that methods that are not synthetic should be traced. *Webster* does not teach or suggest that methods that method can be called by a sufficient scope of one or more other methods should be traced.

Moreover, *Grove*'s discussion on constructing call graphs does not remedy the foregoing deficiencies.

Moreover, there is no reason to modify *Berkley* to arrive at these claim limitations. *Berkley* discloses monitoring classes in an object-oriented environment. Applicants respectfully assert that *Berkley* cannot be modified to trace on a method by method basis without **destroying *Berkley* as a reference**. Stated another way, *Berkley* cannot be modified to trace on a method by method basis without **changing the fundamental principle of operation of *Berkley***. Therefore, at the time of

Applicants' invention, it would not have been obvious to one of ordinary skill in the art to modify *Berkley* to arrive at the claimed invention.

Applicants note that they have taught a fundamentally different technique than any combination of *Berkley*, *Webster*, and *Grove*. Applicant's invention recited in claim 39 is not to use brute force to determine a call graph (which may be inaccurate anyway). Instead, claim 39 recites an elegant solution of "*means for tracing said method for a particular purpose only if said method calls another method, said method can be called by a sufficient scope of one or more other methods, and said method is not a synthetic method.*"

For all of the foregoing reasons, Berkley in view of Webster in further view of Grove fails to disclose, "*means for tracing said method for a particular purpose only if said method calls another method, said method can be called by a sufficient scope of one or more other methods, and said method is not a synthetic method*," as claimed.

Claim 40

Applicants respectfully assert that *Berkley* in view of *Webster* in further view of *Grove* fails to disclose, "*tracing said method for a particular purpose only if said method calls one or more different methods and can be called by a sufficient scope of one or more other methods*," as claimed.

Berkley by itself does not disclose these claim limitations. Claim 40 has the limitation of *methods that call one or more other methods.* Moreover, claim 40 has the additional limitation of *said method can be called by a sufficient scope of one or more other methods.* Note that this results in a far different set of methods being traced than tracing **both** methods that *call one or more other methods* and methods that *can be called by a sufficient scope of one or more other methods.* Applicants have taught a way to **limit** the number of methods traced, while not resorting to complex techniques such as building call graphs.

Moreover, it would not be obvious to trace for methods that both call one or more other methods and can be called by a sufficient scope of one or more other methods, even if calling scope is known in JAVA programming techniques. First of all, even all if methods of a certain scope were to be traced, the claim limitations would not be met. Tracing all methods of sufficient scope would trace those that **do not** call one or more methods. Thus, **too many** methods would be traced.

Furthermore, *Webster* in further view of *Grove* do not remedy these deficiencies. As noted above, *Webster* discloses that a user can provide a file that *lists* methods to trace. *Webster* does not teach or suggest that a method that calls another method should be added to the list of methods to trace. *Webster* does not teach or suggest that methods that method can be called by a sufficient scope of one or more other methods should be traced.

Moreover, *Grove*'s discussion on constructing call graphs does not remedy the foregoing deficiencies.

Moreover, there is no reason to modify *Berkley* to arrive at these claim limitations. *Berkley* discloses monitoring classes in an object-oriented environment. Applicants respectfully asset that *Berkley* cannot be modified to trace on a method by method basis without **destroying *Berkley* as a reference**. Stated another way, *Berkley* cannot be modified to trace on a method by method basis without **changing the fundamental principle of operation of *Berkley***. Therefore, at the time of Applicants' invention, it would not have been obvious to one of ordinary skill in the art to modify *Berkley* to arrive at the claimed invention.

Applicants note that they have taught a fundamentally different technique than any combination of *Berkley*, *Webster*, and *Grove*. Applicant's invention recited in claim 40 is not to use brute force to determine a call graph (which may be inaccurate anyway). Instead, claim 40 recites an elegant solution of "*tracing said method for a particular purpose only if said method calls one or more different methods and can be called by a sufficient scope of one or more other methods.*"

For all of the foregoing reasons, Berkley in view of Webster in further view of Grove fails to disclose, "*tracing said method for a particular purpose only if said method calls one or*

more different methods and can be called by a sufficient scope of one or more other methods," as claimed.

Claim 47

First, Applicants respectfully assert that the Office Action fails to present a *prima facie* case against claim 47. Claim 47 recites the following:

> A process for monitoring, comprising:
> accessing a method;
> automatically determining whether said method is complex, said step of automatically determining includes automatically determining that said method is complex if said method satisfies the following criteria:
> said method calls another method;
> said method has an access level of public or package in the JAVA programming language; and
> said method is not flagged by a compiler as being synthetic; and
> adding a tracer to said method only if said method is automatically determined to be complex.

Applicants respectfully assert that in the rejection to claims 1-3, 5 and 8, the Office Action did not address all limitations of claim 47, **as arranged (or combined)** in claim 47. Applicants respectfully assert that none of claims 1-3, 5, and 8 has claim limitations that mirror the **combination of** limitations from claim 47. For example, only claim 8 (which depends from claim 1) of those claims has claim limitations that pertain to a tracer. However, claim 8 does not have claim limitations that pertain to, "said method has an access level of public or package in the JAVA programming language." Nor does claim limitations that pertain to, "said method is not flagged by a compiler as being synthetic."

Therefore, no reasoning has been supplied in the rejection of claims 1-3, 5, and 8 for arranging all the claim limitations, as combined by Applicants. Consequently, the Office Action fails to present a *prima facie* case of unpatentablity under 35 U.S.C. Section 103.

Applicants further respectfully assert that this failure to address the limitations of claim 47,

as combined by the Applicants, makes it evident that the Examiner is not considering the Applicants' claimed invention as a whole.

Although the Applicants are not required to present any further arguments in response to the rejection to claim 47, the Applicants elect to make the following arguments.

Berkley by itself does not disclose these claim limitations. Claim 47 has the limitation of *methods that call one or more other methods.* Moreover, claim 47 has the additional limitation of *said method has an access level of public or package in the JAVA programming language.* Moreover, claim 47 has the additional limitation of *said method is not flagged by a compiler as being synthetic.* Note that this results in a far different set of methods being traced than tracing **all of** methods that *call one or more other methods* and methods that *has an access level of public or package in the JAVA programming language* and methods are *not flagged by a compiler as being synthetic.* Applicants have taught a way to **limit** the number of methods traced, while not resorting to complex techniques such as building call graphs.

Furthermore, *Webster* in further view of *Grove* do not remedy these deficiencies. As noted above, *Webster* discloses that a user can provide file that *lists* methods to trace. *Webster* does not teach or suggest that a method that calls another method should be added to the list of methods to trace. *Webster* does not teach or suggest that methods that method can be called by a sufficient scope of one or more other methods should be traced.

Moreover, *Grove*'s discussion on constructing call graphs does not remedy the foregoing deficiencies.

Moreover, there is no reason to modify *Berkley* to arrive at these claim limitations. *Berkley* discloses monitoring classes in an object-oriented environment. Applicants respectfully asset that *Berkley* cannot be modified to trace on a method by method basis without **destroying *Berkley* as a reference**. Stated another way, *Berkley* cannot be modified to trace on a method by method basis without **changing the fundamental principle of operation of *Berkley***. Therefore, at the time of Applicants' invention, it would not have been obvious to one of ordinary skill in the art to modify

Berkley to arrive at the claimed invention.

Applicants also note the following reasons why one would not be motivated to modify *Berkley* to arrive at the limitations of currently amended claim 47. *Berkley* is concerned with inserting functions into an existing application executable of an object-oriented computer system without recompiling the executable. *Berkley* modifies runtime configuration settings to add a setting that specifies the function for at least one class of the application executable. *Berkley* runs the application executable using the modified configuration settings, and if it is determined that a function is active for a class then a re-direction stub is created dynamically to implement the function for the class (col. 2, lines 18-29).

Berkley discusses that in an object oriented system it can be difficult to determine the flow of the program. In this discussion, *Berkley* refers to a call graph and the complexities that arise when different methods call each other. Specifically, *Berkley* states that if an error occurred in a "current" method, then the execution of all "previous" methods is important to know the state that an object was in when the current method was called (col. 1, lines 43-62).

What the foregoing clearly indicates is that *Berkley* is simply not interested in determining which methods are synthetic, or which methods have an access level of either public or package in the JAVA programming language, as claimed. Nor would one of ordinary skill in the art have any reason to modify *Berkley* to determine which methods call another method, are not synthetic, and have an access level of either public or package in the JAVA programming language (and add a tracer to only those methods). To do so would not provide *Berkley* with the information such as, "the state on object was in when the current method was called." Thus, *Berkley* would be rendered unsuitable for its intended purpose.

Applicants note that they have taught a fundamentally different technique than any combination of *Berkley*, *Webster*, and *Grove*. Applicant's invention recited in claim 47 is not to use brute force to determine a call graph (which may be inaccurate anyway). Instead, claim 47 recites an elegant solution of "*adding a tracer to said method only if said method is automatically determined to be complex*."

Applicants note that if the tracer were to be added to too many methods, then the performance of the software that contains the methods could be negatively impacted. However, if the tracer is not added to certain methods, then methods that should be traced might not be traced. It can be very challenging to automatically determine which method should be traced and which need not be traced. Applicants respectfully assert that the prior art **provides no guidance** that would suggest to one of ordinary skill in the art to add tracers to only the set of methods defined by the criteria recited in claim 47.

For all of the foregoing reasons, *Berkley* in view of *Webster* in further view of *Grove* fails to disclose, "*adding a tracer to said method only if said method is automatically determined to be complex,*" as claimed.

Rejection of Claims 9, 11, 19, 21-23, 28-30, 32, 37, 38 and 45-46 Under 35 U.S.C. §103(a)

Claims 9, 11, 19, 21-23, 28-30, 32, 37, 38 and 45-46 were rejected under 35 U.S.C. §103(a) as being unpatentable over *Berkley* in view of *Webster* in further view of U.S. Patent No. 6,662,359 to *Berry et al.* (hereinafter "*Berry*"). Apparently, claim 33-35, 37 and 38 also stand rejected under these references. The rejection is respectfully traversed for the following reasons.

Claim 22

> One or more processor readable storage devices having processor readable code embodied on said processor readable storage devices, said processor readable code for programming one or more processors to perform a process comprising:
>
> automatically determining which methods of a set of methods to modify, said step of determining includes automatically determining which methods call one or more other methods and have an access level of either public or package in the JAVA programming language; and
>
> modifying for a particular purpose only those methods that call one or more other methods and have an access level of either public or package in the JAVA programming language.

Berkley in view of *Webster* in further view of *Berry* fails to disclose, "*modifying for a particular purpose only those methods that call one or more other methods and have an access level*

of either public or package in the JAVA programming language," as claimed.

Berkley by itself does not disclose these claim limitations pertaining to *modifying for a particular purpose only those methods that... have an access level of either public or package in the JAVA programming language,*. a claimed. Moreover, claim 22 has the additional limitation of *methods that call one or more other methods.* Thus, *modifying for a particular purpose* is used for said *methods that call one or more other methods* ***and*** *have an access level of either public or package in the JAVA programming language.*

Moreover, it would not be obvious to "*modify for a particular purpose*" those methods that both *have an access level of either public or package in the JAVA programming language* **and** *call one or more other methods*, even access levels are known in JAVA programming techniques. First of all, even if methods having a certain access level were to be traced, the claim limitations would not be met. Tracing all methods having an access level of *public or package in the JAVA programming language* would *modify for a particular purpose* those that **do not** call one or more methods. Thus, too many methods would be *modified for the particular purpose.* Applicants have solved a problem of modifying **too many** methods. Moreover, Applicants have created a solution that is both elegant and simple to implement.

Likewise, *Webster* in further view of *Berry* do not teach these claim limitations. *Webster* provides no teaching or suggestion to put the following into the list of methods, "*methods that call one or more other methods and have an access level of either public or package in the JAVA programming language.*" *Webster* does not in any way teach or suggest either of these, much less the combination. *Webster* does not teach or suggest that a method that calls another method should be added to the list of methods. *Webster* does not teach or suggest that methods that *have an access level of either public or package in the JAVA programming language* should be added to the list.

It does not appear to the Applicants that the Office Action alleges *Berry* discloses these limitations. *Berry* has apparently been used as allegedly disclosing limitations with respect to inserting hooks into JAVA classes (see Office Action, page 22). Therefore, *Berkley* in view of *Webster* in further view of *Berry* does not disclose the claim limitations.

Moreover, there is no reason to modify *Berkley* to arrive at these claim limitations. *Berkley* discloses monitoring classes in an object-oriented environment. Applicants respectfully asset that *Berkley* cannot be modified to trace on a method by method basis without **destroying *Berkley* as a reference**. Stated another way, *Berkley* cannot be modified to trace on a method by method basis without **changing the fundamental principle of operation of *Berkley***. Therefore, at the time of Applicants' invention, it would not have been obvious to one of ordinary skill in the art to modify *Berkley* to arrive at the claimed invention.

Applicants note that they have taught a fundamentally different technique than any combination of *Berkley*, *Webster*, and *Berry*. Applicant's invention recited in claim 22 is not to use brute force to determine a call graph (which may be inaccurate anyway). Instead, claim 22 recites an elegant solution of "*modifying for a particular purpose only those methods that call one or more other methods and have an access level of either public or package in the JAVA programming language*."

For all of the foregoing reasons, *Berkley* in view of *Webster* in further view of *Berry* fails to disclose, "*modifying for a particular purpose only those methods that call one or more other methods and have an access level of either public or package in the JAVA programming language*," as claimed.

Claim 33

Berkley in view of *Webster* in further view of *Berry* fails to disclose, "tracing said method for a particular purpose only if said method calls another method and said access level satisfies the criterion," as claimed.

Applicants have already addressed these claim limitations with respect to *Berkley* in view of *Webster* in their response to claim 1. *Berry* has apparently been used as allegedly disclosing limitations with respect to inserting hooks into JAVA classes (see Office Action, page 22). However, Applicants do not believe that *Berry* is alleged to disclose, "tracing said method for a

particular purpose only if said method calls another method and said access level satisfies the criterion," as claimed.

Therefore, Applicants respectfully assert that claim 33 is patentable over *Berkley* in view of *Webster* in further view of *Berry*.

Dependent claims

The dependent claims not discussed so far are respectfully asserted to be allowable at least by virtue of their dependence on an Independent claim already discussed. The dependent claims contain additional claim limitations that further distinguish over the art. However, the Applicants elect to not discuss these limitations at this time.

Conclusion

Based on the above amendments and these remarks, reconsideration of claims 1-2, 5-13, 17-23, 28-30, 32-35, 37-42, 44-47, 51 and 52 is respectfully requested.

The Examiner's prompt attention to this matter is greatly appreciated. Should further questions remain, the Examiner is invited to contact the undersigned attorney by telephone.

The Commissioner is authorized to charge any underpayment or credit any overpayment to Deposit Account No. 501826 for any matter in connection with this response, including any fee for extension of time, which may be required.

Respectfully submitted,

Date: December 9, 2011

By: /RonaldMPomerenke#43009/
Ronald M. Pomerenke
Reg. No. 43,009

VIERRA MAGEN MARCUS & DENIRO LLP
575 Market Street, Suite 2500
San Francisco, California 94105-4206
Telephone: (415) 369-9660
Facsimile: (415) 369-9665